Getting Secure Software Assurance Knowledge into Conventional Practice

Nancy R. Mead, Software Engineering Institute [vita[1]]

Linda M. Laird, Stevens Institute of Technology [vita[2]]

Dan Shoemaker, University of Detroit Mercy [vita[3]]

2011-08-01

This paper describes three educational initiatives in support of software assurance education. The first project attempted to identify and document any knowledge, from any source, that could be related to the assurance of software. The second initiative focuses on the development of a master of software assurance reference curriculum. The third initiative implements the reference curriculum as two tracks within a Master of Science in Software Engineering program.

Educational Initiatives to Support Software Assurance Priorities

Cybersecurity is an area of international concern. Yet it is well documented that “commonly used software engineering practices [continue to] permit dangerous defects, which let attackers compromise millions of computers every year” [2]. This is the case because “software engineering lacks the rigorous controls needed to [ensure defect-free] products at acceptable cost” [1]. As a result, participants of the Knowledge Transfer Network Workshop in Paris in March 2009 recognized cybersecurity education as part of the information security, privacy, and assurance roadmap vision. They also identified cybersecurity education as one of the workshop’s lines of development [3].

Another example can be found in the U.S. National Strategy to Secure Cyberspace, which includes a specific priority to create a nationwide national cyberspace awareness and training program [4]. That priority recognizes that two of the barriers to the improvement of cybersecurity are “a lack of familiarity, knowledge, and understanding of the issues” and “an inability to find sufficient numbers of adequately trained ... personnel to create and manage secure systems” [2]. One of the priority’s major initiatives is to “foster adequate training and education programs to support the Nation’s cybersecurity needs” [2].

Although we have sufficient knowledge of the practices needed to assure the secure development, sustainment, and acquisition of code, that knowledge is not entering into the profession in any organized way. Dr. Nasir Memon, a professor at the Polytechnic Institute of New York University, reinforced the need for cybersecurity education: “There is a huge demand, and a lot more schools have created programs, but to be honest, we’re still not producing enough students” [6].

The aim of the three initiatives described in this paper is to take the first substantive steps to disseminating the knowledge for secure software assurance into common use. Together these three programs begin the process of ensuring that conventional higher education contributes in a practical way to the worthwhile goal of a more secure software infrastructure.

Getting the Message Out

The traditional means of disseminating knowledge into any society is through formally constituted education, training, and awareness programs [5]. In the secure software domain, however, there is no single, commonly accepted point of reference to “guide the development and integration of education and training content relevant to software assurance” [7]. The U.S. National Strategy recognizes this necessity in both Priority II (A National Cybersecurity Threat and Vulnerability Reduction Program) and Priority III (a

1. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/230-BSI.html (Mead, Nancy)
2. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/1322-BSI.html (Laird, Linda M.)
3. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/689-BSI.html (Shoemaker, Dan)

National Cyberspace Security Awareness and Training Program). Item five in Priority II identifies the need to reduce and remediate software vulnerabilities and Item two in Priority Three identifies the need to foster adequate training and education programs to support the Nation's cybersecurity needs [4].

However, the dilemma with software assurance is that its knowledge elements cut across many disciplines, rather than being focused in a few. In essence, the knowledge base for software assurance spans a range of traditional studies [9]. These include such dissimilar areas as "software engineering, systems engineering, information systems security engineering, safety, security, testing, information assurance, law and project management" [9]. As a result, potentially meaningful software assurance content appears in many different places, and educators in conventional settings teach it in many different ways.

It is clearly unacceptable to approach the teaching and learning process in such a disjointed way. For this reason, it is important to formulate a consolidated view of the body of knowledge for secure software assurance. In particular, a formal effort is needed to integrate "software assurance content ... into the body of knowledge of each contributing discipline" [7, 9]. There are two practical barriers to achieving this level of integration. First, it is not clear what specific knowledge and skills should be taught in each area. Second, there are no validated methods for delivering that knowledge once it has been identified. This paper describes three initiatives that work together to address these two problems.

Initiative One: Formulating and Disseminating Software Assurance Knowledge

Logically, the first step in integrating new knowledge into a conventional learning setting is to identify, relate, and catalogue what is presently available. That was the purpose of a two-year project funded by the U.S. Department of Defense (DoD) and conducted at the University of Detroit Mercy (UDM). This project attempted to identify and document any knowledge, from any source, that could be related to the assurance of software. That knowledge was culled from all of the usual computing disciplines, such as computer science, software engineering, and information systems. The project also incorporated softer knowledge from beyond the strictly technical areas, such as information security, as well as relevant knowledge from the behavioral and social sciences. The knowledge came from many accessible public and private-sector sources.

The product of this study was a knowledge base that documented and categorized all commonly accepted practices, principles, methodologies, and tools for software assurance. The mind map that underlies the categorization is roughly based on the Department of Homeland Security's (DHS) "Software Assurance: A Guide to the Common Body of Knowledge to Produce, Sustain, and Acquire Secure Software" [7]. However, to ensure the validity of the common body of knowledge (CBK) framework, the mind map was fine tuned and subsequently validated by means of a classic Delphi study, as part of the project. To make it as authoritative as possible, the study used a panel of eleven nationally known experts in secure software assurance.

The knowledge base incorporates as many life cycle methodologies and tools for assuring software as could be identified. It also itemizes all related supporting principles and concepts to ensure the security of internally developed and sustained software. The knowledge base also includes any products and services purchased from outside vendors. The knowledge base is evolutionary and inclusive. Thus, as the literature of the field expands or new sources of knowledge are identified, that material will be catalogued and added.

The purpose of the UDM and DoD initiative was not simply to gather knowledge. The goal was to ensure the teaching of secure software topics in all suitable education, training, and awareness settings. In support of that goal, the project packaged the contents of the knowledge base into discrete learning modules. These modules are meant to facilitate the efficient transfer of software assurance knowledge into all relevant teaching and learning settings. As a result, these modules can be incorporated into a wide range of teaching and learning environments. They are appropriate for traditional graduate, undergraduate, community college, and even high school education, as well as for training and awareness applications.

The modules are intended to be stand-alone learning artifacts capable of conveying all of the requisite knowledge for a discrete topic. At a minimum, each module can be delivered in a conventional classroom. However, the modules include supporting material that also allows them to be delivered in a range of

asynchronous and other web-enabled learning environments. The flexibility of the delivery approach facilitates the efficient transfer of new workforce skills and practices to all types of education, training, and awareness applications.

Each module conveys a logical element of software assurance practice. The entire collection of these modules maps to the body of knowledge contained in the knowledge base. Because that knowledge base is structured on the most commonly accepted model for secure software assurance practice, the DHS Common Body of Knowledge [7], this mapping provides precise guidance about where the newly developed instructional content fits within the commonly accepted understanding of the correct elements of practical software assurance work.

These modules were divided into three topic areas based on the CBK: (1) development of secure code, (2) secure sustainment of code, and (3) acquisition of secure code. The results of the Delphi supported the rationale for this partitioning. To ensure that these modules would be free standing and usable in any application, the development of secure code was further decomposed into risk understanding, as represented by various modules devoted to threat modeling, and a series of modules devoted to secure coding methods and techniques. The sustainment process was further decomposed into ethical hacking (as operational testing for vulnerability identification), environmental monitoring and reporting, risk analysis, authorization, change control, and patch management. Finally, secure acquisition was decomposed into acquisition initiation, secure specification, and contract formulation and delivery management.

Each of the teaching modules incorporates a set of conventional learning artifacts, which are easily recognizable to traditional educators. Every module includes (1) a table of learning specifications, (2) presentation slides for each concept contained in the module, (3) an evaluation process, (4) any relevant web-enabled supporting material such as videos, and (5) a model delivery system. Every module also incorporates a validated set of teaching tools. These tools are optimized to ensure the maximum knowledge transfer for all potential teaching settings.

Following development, the project packaged all of this content onto an innovative knowledge transfer device based on the iPad. It allows the project to disseminate the targeted courseware artifacts to classroom teachers in K-12 to higher education settings. The device is called the Software Assurance Mobile Instructional device, or SAMI. SAMI bundles all of the knowledge developed by this project into a single portable platform, which, in addition to providing all required instructional materials, also allows internet access to the contents of the software assurance repository. The advantage of SAMI is that it provides classroom teachers with all of the knowledge and courseware needed to immediately teach topics that might not have been part of their own background or preparation.

Finally, the project performed extensive field trials to validate the courseware and delivery systems. The beta tests examined the appropriateness of the courseware, software tools, and teaching methodologies, and they were conducted at cooperating institutions of the International Cyber Security Education Coalition (ICSEC) as well as sample universities from around the country. The tests evaluated format, concept clarity, usability, comprehension, accuracy, applicability to job competencies, and effectiveness of delivery. The validation included all types of learning environments as well as a range of delivery options. Each content module was installed on site at the cooperating institution and was administered as required by the delivery protocol. As might be imagined, a large amount of data was gathered. The findings were generally favorable and will be reported in later studies.

Initiative Two: A Master of Software Assurance Reference Curriculum

The second initiative focuses on the development of a master of software assurance reference curriculum [11]. This effort was conducted under the leadership of the Software Engineering Institute (SEI) at Carnegie Mellon University, in support of the Department of Homeland Security's National Cyber Security Division. The involvement of SEI is particularly noteworthy because much of the body of knowledge in secure software assurance is derived from software engineering principles and practices. This project specifies a set of topics and the knowledge and requirements necessary to ensure a properly educated software assurance professional. This project differs from the prior initiative in that it is a comprehensive approach

to the definition of the practical body of knowledge whereas the first initiative focused on the content level. This initiative identifies the topics that effective software assurance professionals must be proficient in and structures that set of topics into a comprehensive curriculum. That curriculum contains just those key knowledge elements required to produce a well-educated practitioner.

The curriculum development team included technical staff from the SEI and faculty from a number of universities, both domestic and international. The final report contains the reference curriculum, a glossary of terms, and the guidelines the team used to develop the curriculum, prerequisites, proposed outcomes when a student graduates, curriculum architecture, proposed curricular body of knowledge, and implementation guidelines for the curriculum. A number of existing artifacts, including the software assurance guide to the body of knowledge [7] and the recent Graduate Software Engineering curriculum guidelines [12] as well as the older *SEI Reports on Graduate Software Engineering Education* [13, 14] were inputs to the project.

The project team also referenced the *Guide to the Software Engineering Body of Knowledge* (SWEBOK) [15] as needed to cross-reference their recommendations with the software engineering knowledge that is fundamental to software assurance. Furthermore, to ensure that the reference curriculum was properly reviewed and validated, invited reviewers and the DHS Workforce Education and Training Working Group performed a broad review of the final product. Additionally, some key industry managers and practitioners generously agreed to be surveyed to further enhance the project team's understanding of the necessary outcomes. To ensure a sufficient level of understanding for implementation purposes, the curriculum also includes a detailed list of knowledge units and the corresponding Bloom's taxonomy levels [16]. A sample of the curriculum body of knowledge appears in Table I. IEEE and ACM have recognized this curriculum as being appropriate for a master's program in software assurance.

Knowledge Area		Bloom Level
1. Assurance Across Life Cycles	1.1. Software Life Cycle Processes	--
	1.1.2. New development	C
	1.1.3. Integration, assembly, and deployment	C
	1.1.4. Operation and evolution	C
	1.1.5. Acquisition, supply, and service	C
	1.2. Software Assurance Processes and Practices	--
	1.2.1. Process and practice assessment	AP
	1.2.2. Software assurance integration into software development life cycle (SDLC) phases	AP
2. Risk Management	2.1. Risk Management Concepts	--
	2.1.1. Types and classification	C
	2.1.2. Probability, impact, severity	C
	2.1.3. Models, processes, metrics	C
	2.2. Risk Management Process	--
	2.2.1. Identification	AP
	2.2.2. Analysis	AP

	2.2.3. Planning	AP
	2.2.4. Monitoring and management	AP
	2.3. Software Assurance Risk Management	--
	2.3.1. Vulnerability and threat identification	AP
	2.3.2. Analysis of software assurance risks	AP
	2.3.3. Software assurance risk mitigation	AP
	2.3.4. Assessment of software assurance processes and practices	AP

Table I. Sample of MSwA 2010 Core Body of Knowledge

Establishment of a new degree program is a very ambitious undertaking. As a consequence, the project team anticipated that some universities would elect to establish tracks or specializations in software assurance within existing master's degree programs, such as in Master of Software Engineering degrees, rather than establish a separate, new degree program. Accordingly, the final report on the curriculum provides guidance on how to implement a track or specialization. The project team developed sample course syllabi, a master bibliography, a workshop available in a virtual training environment, and a podcast. The project team has begun to identify available course material. All project materials are available at http://www.cert.org/mswa/. Current activities focus on transition and adoption of the curriculum recommendations. A software assurance education discussion group has been established on LinkedIn, and focused mentoring is available to universities that wish to establish a software assurance degree program or track. In addition to the Master of Software Assurance reference curriculum, this project produced a set of sample software assurance course outlines for the undergraduate level [17]. These courses might form an area of concentration within a computer science or software engineering undergraduate degree program. The project is now working on a set of sample software assurance course outlines at the community college level.

Initiative Three: Implementing a Practical Software Assurance Curriculum

As a proof of concept, Stevens Institute of Technology is implementing the software assurance reference curriculum, described above, as two tracks within their Master of Science in Software Engineering program. In addition, they are offering two graduate certificates based on the courses in that curriculum. As previously mentioned, it is easier for universities to establish tracks within existing programs than to create entire new programs. In this case, Stevens has three relevant graduate programs: software engineering, systems security engineering, and computer science, each of which contained some of the material from the reference curriculum for software assurance. In addition, the architecture of their Software Engineering Program is extremely flexible, which facilitates adding new certificates and concentrations.

The Stevens software engineering faculty believes that every Stevens software engineering student should know how to engineer and build trusted systems, which includes software assurance. Consequently, they are integrating the software assurance curriculum into the existing software engineering curriculum to the maximum extent possible.

The faculty has encountered several issues with utilizing this curriculum implementation strategy. First, the majority of the software engineering faculty are not particularly strong in security, so even though they are motivated and experienced, they must put forth considerable effort to learn, fully understand, and prioritize the content, as well as to create and remove significant amounts of material. Second, the recommendations of the reference curriculum do not simply map onto the existing Stevens courses. Third, although there are excellent security courses in the computer science department, the required material is spread throughout

multiple courses that have multiple prerequisites. Finally, portions of the software assurance curriculum overlap significantly with two courses in the systems security engineering program.

These issues are being resolved by:

1. taking a phased implementation approach to the curriculum. The material is being rolled into the curriculum over a one-year period and is targeted to be completed by the fall of 2011. The new material will be made available online, on an as-needed basis, to students who take the courses before fall 2011.
2. condensing the required secure development material from the multiple computer science courses into one course specifically designed and offered for the software assurance curriculum.
3. having one of the original reference curriculum authors (a Stevens faculty member) lead the mapping of the reference curriculum to Stevens curriculum. A sample of the mapping appears in Table II.
4. adding more material to two systems security engineering courses and including them in the software assurance tracks.

	Stevens Courses				
Core Topics	53 3	54 0	55 6	56 4	...
1.1 Software Life Cycle Processes		•			
1.2 Software Assurance Processes and Practices		• +			
...					
6.2 Assured Software Development			*	• +	
...					

Table II. Sample Mapping of the Reference Curriculum to Stevens Curriculum

One example of the type and extent of curriculum changes that are occurring is SSW 689: Software Reliability and Safety Engineering. It has expanded to become SSW 689: Engineering of Trusted Systems. This change is a natural evolution and expansion of the original course and course objectives, although it would not have occurred so quickly without the MSwA implementation. Content changes include adding and extending material on trusted systems properties, trusted system architectures and patterns, trust cases, assurance maturity models, threat modeling, misuse and abuse cases, and risk mitigation frameworks. Decreased attention is now given to the variety and detail of reliability models and advanced topics in reliability testing.

The resulting software assurance program at Stevens consists of two new, multi-disciplinary, graduate software assurance tracks within software engineering: one intended for students interested in careers in software development of trusted systems, and one for students interested in careers in acquisition and management of trusted systems. Both tracks share the same six core courses in software engineering, but each has four different, additional required courses. This course requirement differs from the software engineering master's program (without a software assurance track), which consists of the same six core courses and four additional electives.

At Stevens, the program is architected so that students who already have a graduate degree, or who are not yet ready to enroll in a full master's-level program, may take courses to earn a graduate certificate. There are two software assurance certificates, one for students interested in software development of trusted systems,

and one for students interested in acquisition and management of trusted systems. All of these courses may be applied toward a graduate degree.

Of course, students in the software engineering master's program may choose instead to pursue other certificates, such a systems architecture or financial systems concentration. Nevertheless, due to the program's architecture, they will have a stronger foundation in software assurance and trusted systems engineering.

Many of the graduate students at Stevens are practicing software development professionals. To accommodate their schedules, Stevens offers all of these courses in three different formats: (1) traditional classroom, (2) asynchronous online and (3) intensive on site. The last format consists of five full days of classroom instruction at a corporate or government site followed by ten weeks of individual and team assignments conducted online. Additional details of the software assurance and software engineering programs at Stevens can be found at www.stevens.edu/software[5].

Results of this first implementation of the software assurance curriculum will be shared with other schools through traditional dissemination and special mentoring arrangements.

Summary and Conclusions

Our understanding of the knowledge needed to ensure capable software assurance professionals is growing as we work through this process of defining and implementing a curriculum and knowledge base. What is needed now is the ability to popularize that knowledge. Because of the nature of the emerging threats in cyberspace, the profession as a whole is being asked to change in ways that have never been required in the past. To ensure our security, software professionals will have to learn how to develop, sustain, and acquire code in a way that will essentially guarantee freedom from exploitable defects. Moreover, to be of any value, this adjustment will have to take place in an outrageously short period of time. Given the critical importance of secure software to the national interest, the three initiatives described in this paper are designed to work together to advance the process. Each project is beginning to establish the foundation for moving software assurance, which has heretofore been poorly understood and poorly recognized, into the mainstream of education, training, and awareness.

The intent of the reference curriculum project at Carnegie Mellon University is to foster software assurance master's programs and tracks that will teach an explicit curriculum of knowledge and skills necessary for producing well-educated software assurance professionals. The initiative at Stevens is putting the recommendations of the reference curriculum into everyday practice. And every instructor of a computer-related discipline in the project at Detroit Mercy will have access to validated content and instructional materials that can be easily incorporated into currently existing courses. This is particularly important because traditional educators in the target disciplines are not knowledgeable about the requisite practices.

All of these initiatives clarify the boundaries and elements of the teaching and learning process for software assurance education. These three projects are initial steps in the long road to assuring the correctness and integrity of developed software with total confidence. Their contents create a direction and foundation that can be built on for the future of the profession.

References

[1] K. M. Goertzel, T. Winograd, H. L. McKinley, L. Oh, M. Colon, T. McGibbon, E. Fedchak, and R. Vienneau, Software Security Assurance: State-of-the-Art Report (SOAR). U.S. Information Assurance Technology Analysis Center (IATAC) and Data and Analysis Center for Software (DACS), July

5. http://www.stevens.edu/software

2007. Electronic Publication: http://iac.dtic.mil/iatac/download/security.pdf

[2] President's Information Technology Advisory Committee, Cybersecurity: A Crisis of Prioritization. Arlington, VA: Executive Office of the President, National Coordination Office for Information Technology Research and Development, 2005.

[3] Leaders in Security, "Building In ... Information Security, Privacy And Assurance," Paper presented at the Knowledge Transfer Network Paris Information Security Workshop, Paris, France, March 2009.

[4] U.S. Department of Homeland Security, The National Strategy to Secure Cyberspace. Washington, DC: U.S. Department of Homeland Security,2003. http://www.dhs.gov/xlibrary/assets/National_Cyberspace_Strategy.pdf

[5] D. Cogburn, Globalization, Knowledge, Education and Training in the Information Age, United Nations Educational, Scientific and Cultural Organization. Paper presented at the Second International Congress on Ethical, Legal and Societal Challenges of Cyberspace, Monte-Carlo, October 1998. Electronic Publication: http://www.unesco.org/webworld/infoethics_2/eng/papers/paper_23.htm

[6] C. Drew, "Wanted: 'Cyber ninjas,'" *New York Times*, December 29, 2009. Electronic Publication: http://www.nytimes.com/2010/01/03/education/edlife/03cybersecurity.html?emc=eta1

[7] S. T. Redwine, Jr., ed. Software Assurance: A Guide to the Common Body of Knowledge to Produce, Acquire, and Sustain Secure Software, Version 1.1. Washington: U.S. DHS, 2006.

[8] Coverity, Quality problems cost software companies up to $22 million annually according to new report, August 2008. Electronic Publication: http://www.coverity.com/html/press_story65_08_04_08.html

[9] D. Shoemaker, A. Drommi, J. Ingalsbe, and N. R. Mead. "A Comparison of the Software Assurance Common Body of Knowledge to Common Curricular Standards," 20th Conference on Software Engineering Education and Training, Dublin, 2007.

[10] M. Newman, Software Errors Cost U.S. Economy $59.5 Billion Annually. Gaithersburg, MD: National Institute of Standards and Technology (NIST), 2002.

[11] N. R. Mead et al., Software Assurance Curriculum Project Volume I: Master of Software Assurance Reference Curriculum (CMU/SEI-2010-TR-005/

ESD-TR-2010-005). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2010.

[12] Stevens Institute of Technology, Graduate Software Engineering 2009 (GSWe2009) Curriculum Guidelines for Graduate Degree Programs in Software Engineering, 2009. Electronic Publication: http://www.gswe2009.org/

[13] G. Ford, 1991 SEI Report on Graduate Software Engineering Education (CMU/SEI-91-TR-002). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1991. Electronic Publication: http://www.sei.cmu.edu/reports/91tr002.pdf

[14] M. Ardis and G. Ford. 1989 SEI Report on Graduate Software Engineering Education (CMU/SEI-89-TR-21). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1989. Electronic Publicatin: http://www.sei.cmu.edu/reports/89tr021.pdf

[15] A. Abran and J. W. Moore, exec. eds., Guide to the Software Engineering Body of Knowledge, 2004 Version. IEEE Computer Society. Electronic Publication: http://www.computer.org/portal/web/swebok

[16] B. S. Bloom, ed., Taxonomy of Educational Objectives: The Classification of Educational Goals: Handbook I: Cognitive Domain. New York: Longmans, 1956.

[17] N. R. Mead, T. B. Hilburn, and R. C. Linger, Software Assurance Curriculum Project Volume II: Undergraduate Course Outlines (CMU/SEI-2010-TR-019, ESC-TR-2010-019). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2010.

Carnegie Mellon University and IEEE Copyright

1. mailto:permission@sei.cmu.edu